Backdoor Sets for DLL Subsolvers

Stefan Szeider

Department of Computer Science
Durham University
DH1 3LE Durham, England, UK

Abstract

We study the parameterized complexity of detecting small *backdoor sets* for instances of the propositional satisfiability problem (SAT). The notion of backdoor sets has been recently introduced by Williams, Gomes, and Selman for explaining the 'heavy-tailed' behavior of backtracking algorithms. If a small backdoor set is found, then the instance can be solved efficiently by the propagation and simplification mechanisms of a SAT solver. Empirical studies indicate that structured SAT instances coming from practical applications have small backdoor sets. We study the worst-case complexity of detecting backdoor sets with respect to the simplification and propagation mechanisms of the classic Davis-Logemann-Loveland (DLL) procedure. We show that the detection of backdoor sets of size bounded by a fixed integer k is of high parameterized complexity. In particular, we determine that this detection problem (and some of its variants) is complete for the parameterized complexity class W[P]. We achieve this result by means of a generalization of a reduction due to Abrahamson, Downey, and Fellows.

Keywords: Satisfiability, unit propagation, pure literal elimination, backdoor sets, parameterized complexity, W[P]-completeness.

1 Introduction

The propositional satisfiability problem (SAT) is the first problem shown to be NP-complete. It holds a central role in the theory of computational complexity and is of practical relevance for applied areas like verification or planning. SAT instances with n variables can be solved by brute force, checking all 2^n truth assignments; no algorithm is known that runs in time $2^{o(n)}$ in the worst case. However, SAT instances arising from applications often impose a "hidden structure" which allow significantly faster SAT decision than by brute force search.

One example of such hidden structure is based on the concept of *backdoor sets* of variables, recently introduced by Williams, Gomes, and Selman [11, 12]. A *weak backdoor set* of a SAT instance is a set B of variables such that for at least one truth assignment to the variables in B, simplifying the instance according to that assignment yields a satisfiable instance that can be decided in polynomial time by a "subsolver." A subsolver is an incomplete polynomial-time algorithm that uses the propagation and simplification mechanisms of a SAT-solver. A

strong backdoor set of a SAT instance is a set B of variables such that for *every* truth assignment to the variables in B, the resulting simplified SAT instance can be decided by the subsolver (exact definitions are given in Sections 2 and 3 below). As reported by Williams, Gomes, and Selman [12], highly structured problem instances have small weak backdoor sets; for example, for a logistics planning benchmark instance with about 7000 variables, a weak backdoor set of size 12 could be found. However, the minimum size of backdoor sets of non-structured instances, like random 3-SAT, appears to be a constant fraction (about 30%) of the total number of variables (Interian [7]). The dependency among the variables of minimal weak backdoor set is studied by Ruan, Kautz, and Horvitz [10]. It is observed that SAT-solvers may heuristically be quite capable of exploiting the existence of small weak backdoor sets in practice, without necessarily identifying the backdoor sets explicitly [12, 10].

In the sequel we address the worst-case time complexity of deciding whether a given SAT instance has a weak or strong backdoor set of size bounded by some integer k. We study this problem with respect to subsolvers of the standard Davis-Logemann-Loveland (DLL) algorithm. That is, subsolvers that are based on *unit propagation* and *pure literal elimination*, or on one of these two principles.

We can detect a weak/strong backdoor set of size at most k by considering all sets B of k or fewer variables of the given instance, and by checking whether one/all of the $2^{|B|}$ assignments to the variables in B yields an instance that can be decided by the subsolver under consideration. Thus a backdoor set can be detected in time $O(2^k n^{k+\alpha})$ where $O(n^\alpha)$ is the worst-case time complexity of the subsolver. However, such a trivial approach becomes impractical for large n even if the parameter k, the maximum size of a backdoor set, is chosen small. In this paper we tackle the question of whether, in general, a small backdoor set can be found significantly faster than by brute force search.

The framework of Parameterized Complexity (Downey and Fellows [5]) provides an excellent framework for studying this question. A parameterized problem is a set $L \subseteq \Sigma^* \times \Sigma^*$ for some fixed alphabet Σ. For a problem instance $(x, k) \in L$, we refer to x as the main part, and to k as the parameter. Typically (and for all problems considered in the sequel), the parameter is a positive integer (presented in unary). XP denotes the class of parameterized problems which can be solved in polynomial time whenever the parameter is considered as a constant; the above considerations show that the detection of a backdoor set is in XP. If a parameterized problem L can be solved in time $O(f(k)n^c)$ where f is any computable function of the parameter and c is a constant (independent from k), then L is called *fixed-parameter tractable*; FPT denotes the class of all fixed-parameter tractable problems. Parameterized complexity classes are defined as equivalence classes of parameterized problems under a certain parameterized reduction. This parameterized reduction is an extension of the polynomial-time many-one reduction where a parameter for one problem maps into a parameter for another. More specifically, a parameterized problem L reduces to a parameterized problem L' if we can transform an instance (x, k) of L into an instance $(x', g(k))$ of L' in time $f(k) \cdot |x|^{O(1)}$ (f, g are arbitrary computable functions), such that (x, k) is a yes-instance of L if and only if $(x', g(k))$ is a yes-instance of L'. The class XP contains a hierarchy of parameterized complexity classes

$$\text{FPT} \subseteq \text{W[1]} \subseteq \text{W[2]} \subseteq \ldots \subseteq \text{W[P]} \subseteq \text{XP}.$$

All inclusions are assumed to be proper; FPT $\neq$ XP is known [5]. The higher a problem is located in this hierarchy, the more unlikely it is fixed-parameter tractable. The canonical W[P]-complete problem is the following (cf. [5]).

WEIGHTED CIRCUIT SATISFIABILITY
Input: A decision circuit D.
Parameter: A positive integer k.
Question: Does D accept an input assignment of weight k?

If a W[P]-complete problem turns out to be fixed-parameter tractable, then the n-variable SAT problem can be solved in time $2^{o(n)}$ (Abrahamson, Downey and Fellows [1]); a recent treatment of the relationship between parameterized complexity classes and SAT upper bounds can be found in Flum and Grohe [6]. The parameterized problem WEIGHTED MONOTONE CIRCUIT SATISFIABILITY arises from WEIGHTED CIRCUIT SATISFIABILITY by restricting the instances to *monotone* circuits. Surprisingly, WEIGHTED MONOTONE CIRCUIT SATISFIABILITY remains W[P]-hard [1, 5]. Furthermore, the problems remain W[P]-complete if we ask for an accepted input assignment of weight *at most* k (see Section 1).

In this paper we completely classify the parameterized complexity of the problem of whether a SAT instance has a weak or strong backdoor set of size not exceeding a parameter k w.r.t. subsolvers that arise from the DLL procedure. In particular, we determine that detection of weak and strong backdoor sets is W[P]-complete for the considered subsolvers. Thus we provide strong theoretical evidence that these problems are not fixed-parameter tractable. We generalize the proof technique used by Abrahamson, Downey, and Fellows [1] for k-INDUCED SATISFIABILITY and other problems by introducing a certain parameterized problem on *cyclic monotone circuits* (see, e.g., Malik [8]). We show that this new problem, CYCLIC MONOTONE CIRCUIT ACTIVATION, is W[P]-complete. Parameterized reductions of this problem provide the base for our W[P]-hardness results. We think that CYCLIC MONOTONE CIRCUIT ACTIVATION is interesting on its own as its W[P]-hardness proof is conceptually simple, and it provides a means for several other W[P]-hardness proofs.

Notation and Preliminaries

We assume an infinite supply of propositional *variables*. A *literal* is a variable x with an assigned parity $\varepsilon \in \{0,1\}$ and is denoted by x^ε. We also write $x = x^1$ and $\overline{x} = x^0$. A set S of literals is *tautological* if it contains both x and $\overline{x}$ for some variable x. A *clause* is a finite non-tautological set of literals. We consider a finite set of clauses as a *CNF formula* (or *formula*, for short). Clauses of size one are called *unit clauses*. The set of variables occurring (negated or unnegated) in a formula F is denoted by $var(F)$. A literal x^ε is a *pure literal* of a formula F if $x \in var(F)$ and no clause of F contains $x^{1-\varepsilon}$.

A *truth assignment* (or *assignment*, for short) is a map $\tau : X_\tau \to \{0,1\}$ defined on some set X_τ of variables. If X_τ is a singleton $\{x\}$ with $\tau(x) = \varepsilon$, then we denote τ simply by $x = \varepsilon$. An assignment τ is *total* for a formula F if $X_\tau = var(F)$. For $x \in X_\tau$ we define $\tau(\overline{x}) = 1 - \tau(x)$. For an assignment τ and a formula F, $F[\tau]$ denotes the formula obtained from F by removing all clauses which contain a literal x with $\tau(x) = 1$ and removing literals y with $\tau(y) = 0$ from the remaining clauses. An assignment τ *satisfies* a formula

F if $F[\tau] = \emptyset$. A formula is *satisfiable* if it is satisfied by some assignment; otherwise it is *unsatisfiable*. Let F be a formula and $(x, \varepsilon) \in var(F) \times \{0, 1\}$. If F contains the unit clause $\{x^\varepsilon\}$ (or x^ε is a pure literal of F), then we say that the assignment $x = \varepsilon$ can be inferred (in one step) by *unit propagation* (or *pure literal elimination*, respectively). If both $x = 0$ and $x = 1$ can be inferred, then F is unsatisfiable (F contains both $\{x\}$ and $\{\overline{x}\}$).

A *decision circuit* (or *circuit*, for short) D is a triple (G, E, λ) where (G, E) is an acyclic digraph (the *underlying digraph* of D) and λ is a mapping from G to $\{\text{AND}, \text{OR}, \text{NOT}\}$. The elements of G are the *gates* and the elements of E are the *lines* of D. A gate $g \in G$ is called $\lambda(g)$-gate. D is *monotone* if it contains no NOT-gates. The *fanin* (*fanout*) of a gate $g \in G$ is its in-degree (out-degree) in the underlying digraph. We assume that NOT-gates have fanin 1 and that AND/OR-gates have fanin at least one. Gates with fanin 2 are *binary* gates. If E contains the line (g, h) then we say that g is a *predecessor of* h and that h is a *successor of* g. Gates with fanin 0 are the *input gates* of the circuit and gates with fanout 0 are the *output gates* of the circuit. We assume that every circuit has exactly one output gate. If the underlying digraph of a circuit D is a tree, then D can be identified with a boolean formula. An *input assignment* ν for a circuit D is a mapping from the set of input gates to $\{0, 1\}$. An input assignment ν propagates through the circuit in the natural way, e.g., for an AND-gate g with predecessors $g_1, \ldots, g_n$, we have $\nu(g) = \min_{i=1}^{n} \nu(g_i)$. A circuit D *accepts* an input assignment ν if $\nu(u) = 1$ holds for the output gate u of D. The *weight* of an input assignment is the number of input gates that are assigned to 1.

Note that a monotone circuit with n input gates accepts an input assignment of weight at most k for some $k \leq n$ if and only if it accepts an input assignment of weight exactly k. If D is non-monotone, then we can still obtain in polynomial time a circuit D' with nk input gates such that D accepts an input assignment of weight at most k if and only if D' accepts an input assignment of weight exactly k (D' can be obtained from D by adding an OR-gate of fanin k in front of each input gate). Furthermore, by means of a standard construction, we can transform a circuit D into a circuit D_2 (D_2 has the same input gates as D) by replacing gates of fanin greater than 2 by several binary gates. The construction of D_2 from D can be carried out in polynomial time, and both circuits accept the same input assignments.

2 Subsolvers

The Davis-Putnam (DP) procedure [4] and the related Davis-Logemann-Loveland (DLL) procedure [3] are certainly the best known complete algorithms for solving the satisfiability problem. Complete state-of-the-art SAT-solvers are typically based on variants of the DLL procedure. A concise description of these procedures can be found in Cook and Mitchell [2]. Both procedures, DP and DLL, search for a satisfying assignment, applying first *unit propagation* and *pure literal elimination* as often as possible. Then, DLL makes a case distinction on the truth value of a variable, and DP eliminates a variable x by replacing the clauses in which x occurs by all the clauses that can be obtained by resolving on x. The DLL procedure is sketched in Fig. 1.

If we use only unit propagation and pure literal elimination, then we get an

Procedure DLL(F)
Input: A CNF formula F.
Output: Either a truth assignment which satisfies F or "unsatisfiable".

1. *Trivial Decision:* If $F = \emptyset$, then return the empty satisfying assignment; if F contains the empty clause, then return "unsatisfiable."
2. *Unit Propagation:* If F contains a unit clause $\{x^\varepsilon\}$, then call DLL($F[x = \varepsilon]$). If a satisfying assignment τ for $F[x = \varepsilon]$ is returned, then return $\tau \cup \{x = \varepsilon\}$; otherwise return "unsatisfiable."
3. *Pure Literal Elimination:* If F contains a pure literal x^ε, then call DLL($F[x = \varepsilon]$). If a satisfying assignment τ for $F[x = \varepsilon]$ is returned, then return $\tau \cup \{x = \varepsilon\}$; otherwise return "unsatisfiable."
4. *Branching:* Choose a variable $x \in var(F)$.
   (a) Call DLL($F[x = 0]$). If a satisfying assignment τ for $F[x = 0]$ is returned, then return $\tau \cup \{x = 0\}$.
   (b) Otherwise, call DLL($F[x = 1]$). If a satisfying assignment τ for $F[x = 1]$ is returned, then return $\tau \cup \{x = 1\}$.
   (c) Otherwise return "unsatisfiable."

Figure 1: The Davis-Logemann-Loveland (DLL) procedure

incomplete algorithm which decides satisfiability for a subclass of CNF formulas. (Whenever the algorithm reaches the branching step, it halts and outputs "give up".) This incomplete algorithm is an example of a "subsolver" as considered by Williams, et al. [11]; a polynomial-time algorithm $\mathcal{S}$ is called a *subsolver* if it either correctly decides satisfiability of the given formula F or it gives up. Moreover, it is required that if the subsolver $\mathcal{S}$ decides that F is satisfiable, it also returns a satisfying assignment, and that $\mathcal{S}$ satisfies the following basic conditions: first, that it decides the empty formula as being satisfiable and a formula containing the empty clause as being unsatisfiable, and second, that if it decides the satisfiability of a formula F, then it does so for $F[x = \varepsilon]$ for any $(x, \varepsilon) \in var(F) \times \{0, 1\}$.

The DLL procedure gives rise to three non-trivial subsolvers: UP+PL (unit propagation and pure literal elimination are available), UP (only unit propagation is available), PL (only pure literal elimination is available).

3 Backdoor Sets

The power of a subsolver can be enhanced by taking an assignment τ to a few variables of the given formula F and inputting $F[\tau]$ to the subsolver. This idea leads to the concept of backdoor sets (cf. [11, 12]).

A set B of variables is a *weak backdoor set* of a formula F w.r.t. a subsolver $\mathcal{S}$ if $B \subseteq var(F)$ and there exists an assignment $\tau : B \to \{0, 1\}$ such that $\mathcal{S}$ returns a satisfying assignment for the input $F[\tau]$; we also say that B is a *weak $\mathcal{S}$-backdoor set*. The set B is a *strong backdoor set* of F w.r.t. $\mathcal{S}$ if $B \subseteq var(F)$ and for every assignment $\tau : B \to \{0, 1\}$, the subsolver $\mathcal{S}$ decides whether $F[\tau]$

is satisfiable or not; we also say that B is a *strong $\mathcal{S}$-backdoor set*.

Similarly one can define backdoor sets with respect to a class $\mathcal{C}$ of formulas where membership in $\mathcal{C}$ and satisfiability of formulas in $\mathcal{C}$ can be decided in polynomial time.

Note that by definition, unsatisfiable formulas do not have weak backdoor sets, and that $B = var(F)$ is always a weak backdoor set of any satisfiable formula F. Moreover, if F is satisfiable, then every strong backdoor set of F is also a weak backdoor set of F w.r.t. any subsolver $\mathcal{S}$, but the converse does not hold in general.

For a subsolver $\mathcal{S}$ we consider the following two parameterized problems.

WEAK $\mathcal{S}$-BACKDOOR
Input: A formula F.
Parameter: A positive integer k.
Question: Does F have a weak $\mathcal{S}$-backdoor set B of size at most k?

STRONG $\mathcal{S}$-BACKDOOR
Input: A formula F.
Parameter: A positive integer k.
Question: Does F have a strong $\mathcal{S}$-backdoor set B of size at most k?

In the next section we formulate an intermediate problem on cyclic monotone circuits which will allow us to determine the complexity of backdoor set detection for the nontrivial subsolvers UP+PL, UP, and PL.

4 Cyclic Monotone Circuits

A *cyclic monotone circuit* is a monotone circuit whose underlying digraph may contain directed cycles. Cyclic circuits have been considered by several authors, see, e.g., Malik [8] for references. We assume that a cyclic monotone circuit may have no input or output gates.

Consider a set A of gates of a cyclic monotone circuit D (we think of the gates in A to be *activated*). The *successor set* $s(A)$ of A contains all gates g of D for which at least one of the following holds:

- $g \in A$;
- g is an AND-gate and all predecessors of g are in A;
- g is an OR-gate and at least one predecessor of g is in A.

If we take iteratively successor sets of A (i.e., we compute a sequence of sets $A^0 \subseteq A^1 \subseteq A^2 \subseteq \ldots$ with $A^0 = A$ and $A^{i+1} = s(A^i)$) then we end up with a set A^* such that $s(A^*) = A^*$. We call A^* the *closure* of the starting set A. Since $A^i \subseteq s(A^i)$ holds always by monotonicity, the closure of A for a cyclic monotone circuit D with n gates is obtained after at most n iterations. We say that A *activates* D if the closure A^* contains all gates of D.

Consider, for example, the cyclic monotone circuit exhibited in Fig. 2. The set $\{g_1\}$ activates the circuit, since we have $s(s(\{g_1\})) = s(\{g_1, g_2\}) = \{g_1, g_2, g_3\}$. However, the set $\{g_2\}$ does not activate the circuit, since $s(\{g_2\}) = \{g_2\} = \{g_2\}^* \neq \{g_1, g_2, g_3\}$.

We are interested in finding a small set of gates that activates a given cyclic monotone circuit. To this end, we define the following parameterized problem.

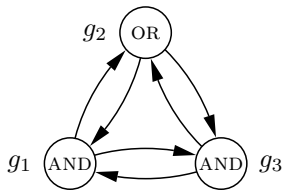

Figure 2: A cyclic monotone circuit.

CYCLIC MONOTONE CIRCUIT ACTIVATION
Instance: A cyclic monotone circuit D.
Parameter: A positive integer k.
Question: Does some starting set A containing at most k gates activate D?

Lemma 1. CYCLIC MONOTONE CIRCUIT ACTIVATION *is* W[P]*-complete. The problem remains* W[P]*-complete for instances without input or output gates.*

Proof. We show membership in W[P] by reducing the problem to WEIGHTED CIRCUIT SATISFIABILITY. Given a cyclic monotone circuit D with n gates, we construct an acyclic monotone circuit C as follows. For every $\circ$-gate g of D, $\circ \in \{\text{AND}, \text{OR}\}$, with predecessors $g_1, \ldots, g_r$ and $0 \leq t \leq n$, we add a gate $g[t]$ to C as follows. For $t = 0$, the gate $g[0]$ is an input gate of C, and for $t > 0$, we put

$$g[t] = g[t-1] \vee (\textstyle\bigcirc_{i=1}^{r} g_i[t-1]).$$

Finally, we add the output gate

$$u = \bigwedge_{g \in D} g[n].$$

It is straightforward to verify that C accepts a weight k input assignment if and only if some starting set of size k activates D. Hence CYCLIC MONOTONE CIRCUIT ACTIVATION is in W[P].

To show W[P]-hardness, we reduce from WEIGHTED MONOTONE CIRCUIT SATISFIABILITY, using ideas from Abrahamson, Downey, and Fellows [1]. Let C be a monotone circuit with n input gates $x_1, \ldots, x_n$ and the output gate u. We construct a cyclic monotone circuit D as follows. We take $k+1$ copies of C, say $C[1], \ldots, C[k+1]$, and denote the copy of a gate g in $C[j]$ by $g[j]$. We add n identical AND-gates $h_1, \ldots, h_n$, each defined by

$$h_i = \bigwedge_{j=1}^{k+1} u[j].$$

We 'feed-back' the gates h_i to the input gates of the circuits $C[1], \ldots, C[k+1]$, adding all the lines $(h_i, x_i[j])$ for $j = 1, \ldots, k+1$ and $i = 1, \ldots, n$. This concludes the construction of D. Observe that D has no input or output gates.

We show that C accepts an input assignment of weight at most k if and only if a starting set of size at most k activates D.

Assume that C accepts an input assignment ν of weight k. We take $A = \{\, h_i : 1 \leq i \leq n,\ \nu(x_i) = 1 \,\}$ and put $A^0 = A$ and $A^i = s(A^{i-1})$ for $i > 0$.

Let d be the length of a longest path in the underlying digraph of C from some input gate x_i to the output gate u (i.e., d is the 'depth' of C). Since C accepts ν, it follows that after $d+1$ iterations all output gates $u[j]$ are activated, i.e., $\{u[1], \ldots, u[k+1]\} \subseteq A^{d+1}$. Hence $\{h_1, \ldots, h_n\} \subseteq A^{d+2}$. In the next step *all* input gates of the circuits $C[i]$ are activated. After d more iterations, going through the circuits $C[i]$ a second time, finally all gates of D belong to $A^{2d+2} = A^*$. Hence A activates D.

Conversely, assume that a starting set A of size at most k activates D, but C accepts no input assignment of weight at most k (we aim to get a contradiction). Since $|A| \leq k$, there must be at least one $C[j]$, $j \in \{1, \ldots, k+1\}$, such that A does not contain any gate of $C[j]$. Since A activates D, $u[j] \in A^*$. Let t be the smallest integer such that $u[j] \in A^t$. Since no gate of $C[j]$ is in the starting set A, some of the input gates of $C[j]$ are activated at some later step such that the activation of the input gates propagates through $C[j]$ to $u[j]$. In other words, we have $X' \subseteq \{x_1[j], \ldots, x_n[j]\} \in A^s$ for some $s < t$ such that $C[j]$ accepts the input assignment ν' of $C[j]$ with $\nu'(x_i[j]) = 1$ if and only if $x_i[j] \in A^s$. By assumption, $|X'| > k$ follows. Consequently, $|\{h_1, \ldots, h_n\} \cap A^s| > k$. This is only possible if all $u[i]$, $1 \leq i \leq n$, are in A^{s-1}. In particular, $u[j] \in A^{s-1}$ and so $t \leq s-1$, a contradiction to $s < t$. Hence C accepts some input assignment of weight at most k. This completes the proof of the lemma. □

It is easy to verify that some starting set of size k activates a cyclic monotone circuit D if and only if some starting set of size k activates the corresponding circuit D_2 that contains only binary gates (see Section 1). Consequently, CYCLIC MONOTONE CIRCUIT ACTIVATION remains W[P]-hard for cyclic monotone circuits that contain only binary gates.

5 Backdoor Sets for Non-trivial Subsolvers

Lemma 2. WEAK $\mathcal{S}$-BACKDOOR *is in* W[P] *for any* $\mathcal{S} \in \{\text{UP+PL}, \text{UP}, \text{PL}\}$.

Proof. We reduce WEAK UP+PL-BACKDOOR to WEIGHTED CIRCUIT SATISFIABILITY. Let F be an instance of UP+PL-BACKDOOR with n variables. We construct an acyclic circuit C with $2n$ input gates that accepts a weight k input assignment if and only if F has a weak UP+PL-backdoor set of size k.

We describe C as consisting of $n+1$ layers, $L_0, \ldots, L_n$. Each layer L_t has input gates $x^0[t]$ and $x^1[t]$ for every $x \in \mathit{var}(F)$. We think of the values of $x^0[t]$ and $x^1[t]$ under some assignment ν as representing the value of the variable x under some assignment τ of F after t propagation steps. That is, $\nu(x^0[t]) = \nu(x^1[t]) = 0$ means that $\tau(x)$ is not defined at step t; $\nu(x^\varepsilon[t]) = 1$ means that $\tau(x) = \varepsilon$ at step t. The construction of C will guarantee that $\nu(x^0[t]) = \nu(x^1[t]) = 1$ cannot be the case for any input assignment ν accepted by C. The input gates of the first layer are the input gates of the whole circuit C. A layer L_t, $t < n$, contains gates that are connected to the input gates of the next layer L_{t+1}. The last layer L_n defines the output gate u of C. Next we describe the construction of C in detail.

For $x \in var(F)$, $\varepsilon \in \{0,1\}$, and $t \in \{0,\ldots,n-1\}$, we put

$$\begin{aligned} x^\varepsilon[t+1] &= x^\varepsilon[t] \vee & (1)\\ &\left(\bigwedge_{C\in F \text{ with } x^{1-\varepsilon}\in C}\left(\bigvee_{y^\eta\in C} y^\eta[t]\right)\right) \vee & (2)\\ &\left(\bigvee_{C\in F \text{ with } x^{\varepsilon}\in C}\left(\bigwedge_{y^\eta\in C\setminus\{x^\varepsilon\}} y^{1-\eta}[t]\right)\right). & (3)\end{aligned}$$

The disjunctive term in (1) ensures that once an assignment to a variable is made it is not changed at a later step. The circuits defined in (2) express pure literal elimination: we set x^ε to 1 at step $t+1$ if all clauses that contain the complementary literal $x^{1-\varepsilon}$ are satisfied at step t. The circuits defined in (3) express unit propagation: we set x^ε to 1 at step $t+1$ if there is some clause in F containing x^ε and all other literals in the clause are set to 0 at step t. It remains to ensure that two input gates $x^\varepsilon[t]$ and $x^{1-\varepsilon}[t]$, representing complementary literals, are never both set to 1, and that finally, at step n, all clauses of F are satisfied. Hence we define the output gate u as

$$u = \left(\bigwedge_{\substack{x\,\in\, var(F)\\ 0\le t\le n}} \neg(x^\varepsilon[t]\wedge x^{1-\varepsilon}[t])\right) \wedge \bigwedge_{C\in F}\bigvee_{y^\eta\in C} y^\eta[n].$$

It is straightforward to verify that C accepts an input assignment of weight k if and only if F has a weak UP+PL-backdoor set of size k. Hence WEAK UP+PL-BACKDOOR is in W[P]. For the problems WEAK UP-BACKDOOR and WEAK PL-BACKDOOR we proceed similarly, omitting the constructions (2) or (3), respectively. □

Lemma 3. STRONG $\mathcal{S}$-BACKDOOR *is in* W[P] *for any* $\mathcal{S} \in \{\text{UP+PL}, \text{UP}, \text{PL}\}$.

Proof. We reduce STRONG UP+PL-BACKDOOR to WEIGHTED CIRCUIT SATISFIABILITY, extending the construction of the proof of Lemma 2. Let F be an instance of STRONG UP+PL-BACKDOOR with n variables. We construct a circuit D with $2^k n$ input gates that accepts a weight $2^k k$ input assignment if and only if F has a strong UP+PL-backdoor set of size k.

For $i = 1,\ldots,2^k$ we construct circuits D_i as in the proof of Lemma 2; each D_i consists of $n+1$ layers and has input gates $x_i^\varepsilon[t]$ for $\varepsilon \in \{0,1\}$, $x \in var(F)$, and $t \in \{0,\ldots,n\}$. The layers of D_i consist of gates as defined in (2) and (3). The output gate u_i of D_i is defined by

$$\begin{aligned} u_i &= \left(\bigwedge_{x\in var(F)} \neg(x^\varepsilon[0]\wedge x^{1-\varepsilon}[0])\right) \wedge\\ &\left(\bigwedge_{C\in F}\bigvee_{y^\eta\in C} y^\eta[n] \vee \bigvee_{C\in F}\bigwedge_{y^\eta\in C} y^{1-\eta}[0] \vee \bigvee_{\substack{x\,\in\, var(F)\\ 1\le t\le n}} (x^\varepsilon[t]\wedge x^{1-\varepsilon}[t])\right).\end{aligned}$$

The difference to the construction in the proof of Lemma 2 is that we also allow the detection of unsatisfiability. We use the fact that unsatisfiability of a formula can be detected by unit propagation and pure literal elimination if and only if the formula contains the empty clause, or both $x = 0$ and $x = 1$ can be inferred.

We combine the circuits $D_1, \ldots, D_{2^k}$ and define the output gate u of D by setting

$$u = \bigwedge_{i=1}^{2^k} u_i \wedge \tag{4}$$

$$\left(\bigwedge_{1 \leq i < j \leq 2^k} \bigvee_{x \in var(F)} x_i^0[0] \not\equiv x_j^0[0] \right) \wedge \tag{5}$$

$$\left(\bigwedge_{\substack{x \in var(F) \\ 1 \leq i < j \leq 2^k}} (x_i^0[0] \vee x_i^1[0]) \equiv (x_j^0[0] \vee x_j^1[0]) \right) \tag{6}$$

where $p \not\equiv q$ abbreviates $(p \wedge \neg q) \vee (\neg p \wedge q)$, and $p \equiv q$ abbreviates $(p \wedge q) \vee (\neg p \wedge \neg q)$. Part (4) ensures that all the circuits D_i accept the input assignment. Part (5) ensures that the input assignment to different copies D_i, D_j, for $i \neq j$, differ in at least one position. Part (6) ensures that all circuits D_i, $1 \leq i \leq 2^k$, receive input assignments that correspond to the same set B of variables of F. We claim that F has a strong UP+PL-backdoor set B of size k if and only if D accepts an input assignment of weight k.

Assume that $B \subseteq var(F)$ is a strong UP+PL-backdoor set of F with $|B| = k$. Let $\{\tau_1, \ldots, \tau_{2^k}\}$ be the set of all assignments $\tau_i : B \to \{0, 1\}$. We define an input assignment ν of D by setting for all $(x, \varepsilon) \in var(F) \times \{0, 1\}$

$$\nu(x_i^\varepsilon[0]) = \begin{cases} 1 & \text{if } x \in B \text{ and } \tau_i(x) = \varepsilon; \\ 0 & \text{otherwise.} \end{cases}$$

We observe that for each D_i, τ sets exactly k input gates to 1, hence the weight of τ is $2^k k$. Since B is a strong UP+PL-backdoor set, it follows by construction of D that D accepts ν.

Conversely, assume that D accepts an input assignment ν of weight $2^k k$. For $i = 1, \ldots, 2^k$ let $B_i = \{\, x \in var(F) : \nu(x_i^0[0]) = 1 \text{ or } \nu(x_i^1[0]) = 1) \,\}$ and define an assignment $\tau_i : B_i \to \{0, 1\}$ such that $\tau_i(x) = 1$ if and only if $\nu(x_i^1[0]) = 1$. Part (6) of the definition of D implies $B_i = B_j$ for all $1 \leq i < j \leq 2^k$, and part (5) implies $|\{\tau_1, \ldots, \tau_{2^k}\}| = 2^k$. Thus $\tau_1, \ldots, \tau_{2^k}$ are all possible assignments for the set $B = B_1 = \ldots = B_{2^k}$. Since D accepts ν, it follows that for every $i \in \{1, \ldots, 2^k\}$, the UP+PL-subsolver decides whether $F[\tau_i]$ is satisfiable or not. In summary, B is a strong UP+PL-backdoor set of size k.

Hence we have shown that STRONG UP+PL-BACKDOOR is in W[P]. This holds as well for STRONG UP-BACKDOOR and STRONG PL-BACKDOOR, as we can modify the above construction by omitting (2) or (3), respectively, in the definitions of the circuits D_i. □

Lemma 4. *The problems* WEAK UP+PL-BACKDOOR *and* WEAK UP-BACKDOOR *are* W[P]*-hard. The problems remain* W[P]*-hard for CNF formulas that have exactly one satisfying total assignment.*

Proof. We reduce CYCLIC MONOTONE CIRCUIT ACTIVATION. Let $D = (G, E, \lambda)$ be a cyclic monotone circuit without input or output gates. We may assume that all gates of D are binary (cf. the discussion at the end of Section 4).

For each gate $g \in G$ we define a set F_g of clauses, and we obtain a formula F by taking the union of all sets F_g with $g \in G$. For an AND-gate $g = x_1 \wedge x_2$, the set F_g contains the clauses

$$\{x_1, y_1\}, \{\overline{x_1}, y_1\}, \{x_1, \overline{y_1}\},$$
$$\{x_2, y_2\}, \{\overline{x_2}, y_2\}, \{x_2, \overline{y_2}\},$$
$$\{\overline{x_1}, \overline{y_1}, \overline{x_2}, \overline{y_2}, g\};$$

the variables y_1, y_2 are new variables not occurring outside of these 7 clauses (we call the variables y_1, y_2 *private*). Similarly, for an OR-gate $g = x_1 \vee x_2$, the set F_g contains the clauses

$$\{x_1, y_1\}, \{\overline{x_1}, y_1\}, \{x_1, \overline{y_1}\},$$
$$\{x_2, y_2\}, \{\overline{x_2}, y_2\}, \{x_2, \overline{y_2}\},$$
$$\{\overline{x_1}, \overline{y_1}, z\}, \{\overline{x_2}, \overline{y_2}, g\};$$

again, y_1, y_2 are private variables. By construction, $G \subseteq \mathit{var}(F)$, and since D has no input gates, $\mathit{var}(F) \setminus G$ is the set of all private variables of F. Evidently, each F_g is satisfied by assigning 1 to all its variables; however, if 0 is assigned to at least one variable, at least one clause of F_g is not satisfied. Hence the assignment τ_1 that sets all variables to 1 is the only satisfying total assignment of F. Consequently, for any subsolver $\mathcal{S}$, a set $B \subseteq \mathit{var}(F)$ is a weak $\mathcal{S}$-backdoor set of F if and only if $\mathcal{S}$ extends the assignment $\tau_0 : B \to \{1\}$ to the satisfying assignment τ_1.

From $y_i = 1$ for a private variable y_i we can infer $x_i = 1$ by means of unit propagation, since the clause $\{x_i, \overline{y_i}\}$ is contained in F. Consequently, if B is a weak UP-backdoor set of F, then replacing private variables y_i of B with x_i, yields a weak UP-backdoor set $B' \subseteq G$ with $|B'| \leq |B|$. Moreover, unit propagation on a set F_g behaves exactly as the activation process on the gate g in D. For example, consider F_g for an AND-gate $g = x_1 \wedge x_2$. By unit propagation, we infer from $x_1 = 1$ and $x_2 = 1$ the assignments $y_1 = 1$ and $y_2 = 1$, and, in turn, $g = 1$. (However, setting $g = 1$ does not propagate 'upward' to y_i or x_i.) Thus, a set B of gates of D activates D if and only if for $\tau_0 : B \to \{1\}$, all clauses of $F[\tau_0]$ can be satisfied using several steps of unit propagation; that is, B is a weak UP-backdoor set of F. Hence we have shown that some starting set of size at most k activates D if and only if F has a weak UP-backdoor set of size at most k. Consequently, W[P]-hardness of WEAK UP-BACKDOOR follows from Lemma 1.

Next we show that W[P]-hardness also holds for WEAK UP+PL-BACKDOOR by proving that every weak UP+PL-backdoor set of F is a weak UP-backdoor set. Consider $\emptyset \neq B \subseteq \mathit{var}(F)$ and $\tau_0 : B \to \{1\}$. First we observe that for any variable $x \in \mathit{var}(F)$, the negative literal $\overline{x}$ cannot be pure in $F[\tau_0]$, since otherwise we could infer $x = 0$ by means of pure literal elimination, but

then $F[\tau_0]$ would be unsatisfiable. Since the circuit D has no output gates, every variable of F occurs as x_i or y_i in some set F_g. However, for every pair of variables x_i, y_i, some F_g contains the binary clauses $\{x_i, \overline{y_i}\}$ and $\{\overline{x_i}, y_i\}$. Thus, for x_i being a pure literal of $F[\tau_0]$, $y_i \in B$ must prevail. Then, however, $F[\tau_0]$ contains the unit clause $\{x_i\}$, and so $x_i = 1$ can be inferred by unit propagation, and pure literal elimination is not needed. Similarly, if y_i is a pure literal of $F[\tau_0]$, then $F[\tau_0]$ contains the unit clause $\{y_i\}$, and again $y_i = 1$ can be inferred by unit propagation. We conclude that pure literal elimination is redundant for $F[\tau_0]$. Thus, it follows by induction on $|var(F) \setminus B|$ that B is a weak UP+PL-backdoor set of F if and only if B is a weak UP-backdoor set of F. Hence WEAK UP+PL-BACKDOOR is W[P]-hard. □

Lemma 5. *The problems* STRONG UP+PL-BACKDOOR *and* STRONG UP-BACKDOOR *are* W[P]*-hard.*

Proof. Let $\mathcal{S} \in \{\text{UP+PL}, \text{UP}\}$. We reduce WEAK $\mathcal{S}$-BACKDOOR. Let F be a formula with exactly one satisfying total assignment τ; w.l.o.g., we assume that τ assigns 1 to each variable of F. We obtain a formula F^* from F by taking for every $x \in var(F)$ a new variable x^* and adding the clauses $\{x, x^*\}$ and $\{x, \overline{x^*}\}$ to F. Note that τ also satisfies F^* and that every satisfying assignment τ^* of F^* extends τ.

We show that F has a weak $\mathcal{S}$-backdoor set of size at most k if and only if F^* has a strong $\mathcal{S}$-backdoor set of size at most k.

Let B be a weak $\mathcal{S}$-backdoor set of F. Thus, with input $F[\tau_0]$, $\tau_0 : B \to \{1\}$, the subsolver $\mathcal{S}$ finds the assignment τ that satisfies F. Since the presence of clauses $\{x, x^*\}$ and $\{x, \overline{x^*}\}$ does not prevent any application of unit propagation or pure literal elimination, the subsolver $\mathcal{S}$ finds the assignment τ also with input $F^*[\tau_0]$. Hence B is a weak $\mathcal{S}$-backdoor set of F^*. The set $B^* = \{\, x^* : x \in B \,\}$ is evidently a weak $\mathcal{S}$-backdoor set of F^* and we have $|B| = |B^*|$. However, B^* is also a strong $\mathcal{S}$-backdoor set of F^*, since, by symmetry, it does not matter whether a variable x^* is set to 0 or set to 1.

Conversely, let B^* be a strong $\mathcal{S}$-backdoor set of F^*. Since F^* is satisfiable, B^* is also a weak $\mathcal{S}$-backdoor set of F^*; thus $\mathcal{S}$ extends $\tau_0^* : B^* \to \{1\}$ to a satisfying assignment of F^*. Since $\{x, \overline{x^*}\} \in F^*$, $x^* = 1$ yields $x = 1$ by unit propagation. Hence we can replace each $x^* \in B^*$ by x and still have a weak $\mathcal{S}$-backdoor set $B := \{\, x \in var(F) : x \in B^* \text{ or } x^* \in B^* \,\}$ with $|B| \leq |B^*|$. Thus, the subsolver $\mathcal{S}$ extends $\tau_0 : B \to \{1\}$ to a satisfying assignment of F^*. The clauses in $F^* \setminus F$ are irrelevant for such extension, since as early as a variable $x \in var(F)$ gets the value 1 under some extension of τ_0, the clauses $\{x, x^*\}$ and $\{x, \overline{x^*}\}$ are removed. Consequently B is also a weak $\mathcal{S}$-backdoor set of F. □

Lemma 6. WEAK PL-BACKDOOR *is* W[P]*-hard and remains* W[P]*-hard for CNF formulas which have exactly one satisfying total assignment.*

Proof. We reduce CYCLIC MONOTONE CIRCUIT ACTIVATION as in Lemma 4. Again, let $D = (G, E, \lambda)$ be a cyclic monotone circuit without input or output gates and where all gates are binary. For each gate $g \in G$ we define a set of clauses F_g, and we obtain a formula F by taking the union of all sets F_g with

$g \in G$. For an AND-gate $g = x_1 \wedge x_2$, the set F_g contains the clauses

$$\{x_1, y_1\}, \{x_1, \overline{y_1}\},$$
$$\{x_2, y_1\}, \{x_2, \overline{y_1}\},$$
$$\{y_1, \overline{g}\};$$

for an OR-gate $g = x_1 \vee x_2$, the set F_g contains the clauses

$$\{x_1, x_2, y_1\},$$
$$\{x_1, x_2, \overline{y_1}\},$$
$$\{y_1, \overline{g}\};$$

the variables y_i are *private* variables. We have $G \subseteq \mathit{var}(F)$, and since D has no input gates, $\mathit{var}(F) \setminus G$ is the set of private variables. We show that F has a weak PL-backdoor set of size at most k if and only if some starting set of size at most k activates D. As in the proof of Lemma 4 it follows from the definition of the sets F_g that the only satisfying total assignment of F sets all variables to 1. Pure literal elimination on F_g behaves exactly as the activation process on the corresponding gate: e.g., for an AND-gate $g = x_1 \wedge x_2$, if $\tau_0(x_1) = \tau_0(x_2) = 1$, then the clauses $\{x_1, y_1\}$, $\{x_1, \overline{y_1}\}$, $\{x_2, y_1\}$, and $\{x_2, \overline{y_1}\}$ are removed from the formula and g becomes a pure literal, thus $g = 1$ follows. Hence a set $B \subseteq G$ of gates activates D if and only if B is a weak PL-backdoor set of F. By replacing private variables y_i by x_i, we can find for every weak PL-backdoor set B of F a weak PL-backdoor set $B' \subseteq G$ with $|B'| \leq |B|$. Hence F has a weak PL-backdoor set of size at most k if and only if some starting set of size at most k activates D. Thus we have reduced CYCLIC MONOTONE CIRCUIT ACTIVATION to WEAK PL-BACKDOOR, and the lemma follows. □

Lemma 7. *The problem* STRONG PL-BACKDOOR *is* W[P]*-hard.*

Proof. We reduce WEAK PL-BACKDOOR. Let F be a formula with exactly one satisfying total assignment τ; w.l.o.g., we assume that τ assigns 1 to each variable of F. We obtain a formula F^* from F by adding the unit clause $\{x\}$ for every variable x of F; i.e.,

$$F^* = F \cup \{\, \{x\} : x \in \mathit{var}(F) \,\}.$$

Evidently, τ is also the unique satisfying total assignment of F^*. Let $\emptyset \neq B \subseteq \mathit{var}(F)$ and $\tau_0 : B \to \{1\}$. We observe that a variable is pure in $F[\tau_0]$ if and only if it is pure in $F^*[\tau_0]$. Hence, it follows by induction on $|\mathit{var}(F) \setminus B|$ that B is a weak PL-backdoor set of F if and only if B is a weak PL-backdoor set of F^*. On the other hand, let $\tau_0' : B \to \{0, 1\}$ be any assignment different from τ_0. There is at least one $x \in \mathit{var}(F)$ such that $\tau_0'(x) = 0$. Since $\{x\} \in F^*$, $F^*[\tau_0']$ contains the empty clause, and so the unsatisfiability of $F^*[\tau_0']$ can be decided by any subsolver. Thus, if B is a weak PL-backdoor set of F, B is also a strong PL-backdoor set of F. Since F^* is satisfiable, every strong PL-backdoor set of F^* is also a weak PL-backdoor set of F^*. In summary, F has a weak PL-backdoor set of size at most k if and only if F^* has a strong PL-backdoor set of size at most k. Hence W[P]-hardness of STRONG PL-BACKDOOR follows from Lemma 6. □

In view of the above lemmas we conclude that all the considered problems are W[P]-complete.

Theorem 1. *The problems* WEAK $\mathcal{S}$-BACKDOOR *and* STRONG $\mathcal{S}$-BACKDOOR *are* W[P]*-complete for each subsolver* $\mathcal{S} \in \{\text{UP+PL}, \text{UP}, \text{PL}\}$.

6 Final Remarks

In this paper we have determined the parameterized complexity of the backdoor set detection problem for subsolvers that arise from the DLL/DP procedures. Our results indicate that these problems are computationally hard; it is very unlikely that, in the worst case, smallest backdoor sets for DLL subsolvers can be found more efficiently than by brute force search. Complementary to the findings of the present paper are the results of Nishimura, Ragde, and Szeider [9] on the parameterized complexity of backdoor set detection with respect to the syntactically defined classes HORN and 2-CNF. It turns out that, although weak backdoor set detection with respect to these classes is W[2]-hard, the detection of strong backdoor sets is fixed-parameter tractable! The identification of further polynomial-time classes of SAT instances that allow fixed-parameter tractable backdoor set detection is a challenging new direction of research. For example, it would be interesting to know whether the detection of strong backdoor sets w.r.t. the class RHORN of renamable Horn formulas is fixed-parameter tractable. It is well known that RHORN properly contains the class of all Horn formulas, and RHORN is itself a proper subclass of the class of formulas decidable by unit propagation.

References

[1] K. A. Abrahamson, R. G. Downey, and M. R. Fellows. Fixed-parameter tractability and completeness. IV. On completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, 1995.

[2] S. A. Cook and D. G. Mitchell. Finding hard instances of the satisfiability problem: a survey. In *Satisfiability problem: theory and applications (Piscataway, NJ, 1996)*, pages 1–17. American Mathematical Society, 1997.

[3] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Comm. ACM*, 5:394–397, 1962.

[4] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[5] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer Verlag, 1999.

[6] J. Flum and M. Grohe. Parameterized complexity and subexponential time. *Bulletin of the European Association for Theoretical Computer Science*, 84:71–100, 2004.

[7] Y. Interian. Backdoor sets for random 3-SAT. In *Sixth International Conference on Theory and Applications of Satisfiability Testing, S. Margherita*

Ligure, Portofino, Italy, May 5–8, 2003, (SAT 2003), informal proceedings, pages 231–238, 2003.

[8] S. Malik. Analysis of cyclic combinatorial circuits. *IEEE Transactions on Computer Aided Design*, 13(7):950–956, 1994.

[9] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In H. Hoos and D. G. Mitchell, editors, *Seventh International Conference on Theory and Applications of Satisfiability Testing, 10–13 May, 2004, Vancouver, BC, Canada (SAT 2004), informal proceedings*, pages 96–103, 2004.

[10] Y. Ruan, H. A. Kautz, and E. Horvitz. The backdoor key: A path to understanding problem hardness. In D. L. McGuinness and G. Ferguson, editors, *Proceedings of the 19th National Conference on Artificial Intelligence, 16th Conference on Innovative Applications of Artificial Intelligence*, pages 124–130. AAAI Press / The MIT Press, 2004.

[11] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003*, pages 1173–1178. Morgan Kaufmann, 2003.

[12] R. Williams, C. Gomes, and B. Selman. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. In *Sixth International Conference on Theory and Applications of Satisfiability Testing, S. Margherita Ligure, Portofino, Italy, May 5-8, 2003 (SAT 2003), informal proceedings*, pages 222–230, 2003.